Wikipedia
The Free Encyclopedia

JPEG

JPEG (/ˈdʒeɪpɛɡ/ *JAY-peg*, short for **Joint Photographic Experts Group** and sometimes retroactively referred to as **JPEG 1**)[2][3] is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography. The degree of compression can be adjusted, allowing a selectable trade off between storage size and image quality. JPEG typically achieves 10:1 compression with noticeable, but widely agreed to be acceptable perceptible loss in image quality.[4] Since its introduction in 1992, JPEG has been the most widely used image compression standard in the world,[5][6] and the most widely used digital image format, with several billion JPEG images produced every day as of 2015.[7]

The Joint Photographic Experts Group created the standard in 1992,[8] based on the discrete cosine transform (DCT) algorithm.[9][10] JPEG was largely responsible for the proliferation of digital images and digital photos across the Internet and later social media.[11] JPEG compression is used in a number of image file formats. JPEG/Exif is the most common image format used by digital cameras and other photographic image capture devices; along with JPEG/JFIF, it is the most common format for storing and transmitting photographic images on the World Wide Web.[12] These format variations are often not distinguished and are simply called JPEG.

The MIME media type for JPEG is "image/jpeg", except in older Internet Explorer versions, which provide a MIME type of "image/pjpeg" when uploading JPEG images.[13] JPEG files usually have a filename extension of "jpg" or "jpeg". JPEG/JFIF supports a maximum image size of 65,535×65,535 pixels,[14] hence up to 4 gigapixels for an aspect ratio of 1:1. In 2000, the JPEG group introduced a format intended to be a successor, JPEG 2000, but it was unable to replace the original JPEG as the dominant image standard.[15]

History

Background

The original JPEG specification published in 1992 implements processes from various earlier research papers and patents cited by the CCITT (now ITU-T) and Joint Photographic Experts Group.[1]

The basis for JPEG's lossy compression algorithm is the discrete cosine transform (DCT),[9][10] which was first proposed by Nasir Ahmed as an image compression technique in 1972.[16][10] Ahmed published the DCT algorithm with T. Natarajan and K. R. Rao in a 1974 paper,[17] which is cited in the JPEG specification.[9]

The JPEG specification cites patents from several companies. The following patents provided the basis for its arithmetic coding algorithm.[1]

- IBM

- U.S. patent 4,652,856 (https://patents.google.com/patent/US4652856) – February 4, 1986 – Kottappuram M. A. Mohiuddin and Jorma J. Rissanen – Multiplication-free multi-alphabet arithmetic code
- U.S. patent 4,905,297 (https://patents.google.com/patent/US4905297) – February 27, 1990 – G. Langdon, J. L. Mitchell, W. B. Pennebaker, and Jorma J. Rissanen – Arithmetic coding encoder and decoder system
- U.S. patent 4,935,882 (https://patents.google.com/patent/US4935882) – June 19, 1990 – W. B. Pennebaker and J. L. Mitchell – Probability adaptation for arithmetic coders

- Mitsubishi Electric
  - JP H02202267 (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=JPH02202267) (1021672 (https://patents.google.com/patent/JPH02202267A)) – January 21, 1989 – Toshihiro Kimura, Shigenori Kino, Fumitaka Ono, Masayuki Yoshida – Coding system
  - JP H03247123 (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=JPH03247123) (2-46275 (https://patents.google.com/patent/JPH0834434B2/en)) – February 26, 1990 – Tomohiro Kimura, Shigenori Kino, Fumitaka Ono, and Masayuki Yoshida – Coding apparatus and coding method

The JPEG specification also cites three other patents from IBM. Other companies cited as patent holders include AT&T (two patents) and Canon Inc.[1] Absent from the list is U.S. patent 4,698,672 (https://patents.google.com/patent/US4698672), filed by Compression Labs' Wen-Hsiung Chen and Daniel J. Klenke in October 1986. The patent describes a DCT-based image compression algorithm, and would later be a cause of controversy in 2002 (see *Patent controversy* below).[18] However, the JPEG specification did cite two earlier research papers by Wen-Hsiung Chen, published in 1977 and 1984.[1]

JPEG

JPEG

A photo of a European wildcat with the compression rate, and associated losses, decreasing from left to right

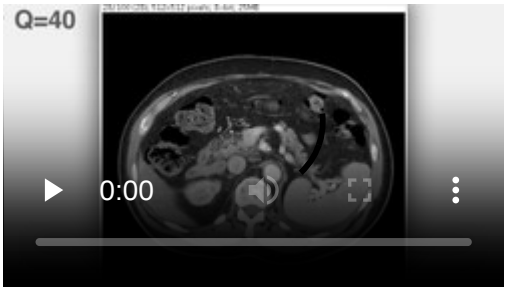
Filename extension	`.jpg`, `.jpeg`, `.jpe` `.jif`, `.jfif`, `.jfi`
Internet media type	`image/jpeg`
Type code	`JPEG`
Uniform Type Identifier (UTI)	public.jpeg
Magic number	`ff d8 ff`
Developed by	Joint Photographic Experts Group, IBM, Mitsubishi Electric, AT&T, Canon Inc.[1]
Initial release	September 18, 1992
Type of format	Lossy image compression format
Extended to	JPEG 2000
Standard	ISO/IEC 10918, ITU-T T.81, ITU-T T.83, ITU-T T.84, ITU-T T.86
Website	jpeg.org/jpeg/ (https://jpeg.org/jpeg/)

JPEG standard

"JPEG" stands for Joint Photographic Experts Group, the name of the committee that created the JPEG standard and other still picture coding standards. The "Joint" stood for ISO TC97 WG8 and CCITT SGVIII. Founded in 1986, the group developed the JPEG standard during the late 1980s. The group

published the JPEG standard in 1992.[5]

In 1987, ISO TC 97 became ISO/IEC JTC 1 and, in 1992, CCITT became ITU-T. Currently on the JTC1 side, JPEG is one of two sub-groups of ISO/IEC Joint Technical Committee 1, Subcommittee 29, Working Group 1 (ISO/IEC JTC 1/SC 29/WG 1) – titled as *Coding of still pictures*.[19][20][21] On the ITU-T side, ITU-T SG16 is the respective body. The original JPEG Group was organized in 1986,[22] issuing the first JPEG standard in 1992, which was approved in September 1992 as **ITU-T Recommendation T.81**[23] and, in 1994, as **ISO/IEC 10918-1**.



Continuously varied JPEG compression (between Q=100 and Q=1) for an abdominal CT scan

The JPEG standard specifies the codec, which defines how an image is compressed into a stream of bytes and decompressed back into an image, but not the file format used to contain that stream.[24] The Exif and JFIF standards define the commonly used file formats for interchange of JPEG-compressed images.

JPEG standards are formally named as *Information technology – Digital compression and coding of continuous-tone still images*. ISO/IEC 10918 consists of the following parts:

Digital compression and coding of continuous-tone still images – Parts[20][22][25]

Part	ISO/IEC standard	ITU-T Rec.	First public release date	Latest amendment	Title	Description
Part 1	ISO/IEC 10918-1:1994 (https://www.iso.org/standard/18902.html)	T.81 (09/92) (http://www.itu.int/rec/T-REC-T.81)	Sep 18, 1992		Requirements and guidelines	
Part 2	ISO/IEC 10918-2:1995 (https://www.iso.org/standard/20689.html)	T.83 (11/94) (http://www.itu.int/rec/T-REC-T.83)	Nov 11, 1994		Compliance testing	Rules and checks for software conformance (to Part 1).
Part 3	ISO/IEC 10918-3:1997 (https://www.iso.org/standard/25037.html)	T.84 (07/96) (http://www.itu.int/rec/T-REC-T.84)	Jul 3, 1996	Apr 1, 1999	Extensions	Set of extensions to improve the Part 1, including the **Still Picture Interchange File Format** (SPIFF).[26]
Part 4	ISO/IEC 10918-4:2024 (https://www.iso.org/standard/85634.html)	T.86 (06/98) (http://www.itu.int/rec/T-REC-T.86)	Jun 18, 1998		Appn Markers	methods for registering some of the parameters used to extend JPEG
Part 5	ISO/IEC 10918-5:2013 (https://www.iso.org/standard/54989.html)	T.871 (05/11) (http://www.itu.int/rec/T-REC-T.871)	May 14, 2011		JPEG File Interchange Format (JFIF)	A popular format which has been the de facto file format for images encoded by the JPEG standard. In 2009, the JPEG Committee formally established an Ad Hoc Group to standardize JFIF as JPEG Part 5.[27]
Part 6	ISO/IEC 10918-6:2013 (https://www.iso.org/standard/59634.html)	T.872 (06/12) (http://www.itu.int/rec/T-REC-T.872)	Jun 2012		Application to printing systems	Specifies a subset of features and application tools for the interchange of images encoded according to the ISO/IEC 10918-1 for printing.

Part 7	ISO/IEC 10918-7:2023 (https://www.iso.org/standard/85635.html)	T.873 (06/21) (http://www.itu.int/rec/T-REC-T.873)	May 2019	November 2023	Reference Software	Provides reference implementations of the JPEG core coding system

Ecma International TR/98 specifies the JPEG File Interchange Format (JFIF); the first edition was published in June 2009.[28]

Patent controversy

In 2002, Forgent Networks asserted that it owned and would enforce patent rights on the JPEG technology, arising from a patent that had been filed on October 27, 1986, and granted on October 6, 1987: U.S. patent 4,698,672 (https://patents.google.com/patent/US4698672) by Compression Labs' Wen-Hsiung Chen and Daniel J. Klenke.[18][29] While Forgent did not own Compression Labs at the time, Chen later sold Compression Labs to Forgent, before Chen went on to work for Cisco. This led to Forgent acquiring ownership over the patent.[18] Forgent's 2002 announcement created a furor reminiscent of Unisys' attempts to assert its rights over the GIF image compression standard.

The JPEG committee investigated the patent claims in 2002 and were of the opinion that they were invalidated by prior art,[30] a view shared by various experts.[18][31]

Between 2002 and 2004, Forgent was able to obtain about US$105 million by licensing their patent to some 30 companies. In April 2004, Forgent sued 31 other companies to enforce further license payments. In July of the same year, a consortium of 21 large computer companies filed a countersuit, with the goal of invalidating the patent. In addition, Microsoft launched a separate lawsuit against Forgent in April 2005.[32] In February 2006, the United States Patent and Trademark Office agreed to re-examine Forgent's JPEG patent at the request of the Public Patent Foundation.[33] On May 26, 2006, the USPTO found the patent invalid based on prior art. The USPTO also found that Forgent knew about the prior art, yet it intentionally avoided telling the Patent Office. This makes any appeal to reinstate the patent highly unlikely to succeed.[34]

Forgent also possesses a similar patent granted by the European Patent Office in 1994, though it is unclear how enforceable it is.[35]

As of October 27, 2006, the U.S. patent's 20-year term appears to have expired, and in November 2006, Forgent agreed to abandon enforcement of patent claims against use of the JPEG standard.[36]

The JPEG committee has as one of its explicit goals that their standards (in particular their baseline methods) be implementable without payment of license fees, and they have secured appropriate license rights for their JPEG 2000 standard from over 20 large organizations.

Beginning in August 2007, another company, Global Patent Holdings, LLC claimed that its patent (U.S. patent 5,253,341 (https://patents.google.com/patent/US5253341)) issued in 1993, is infringed by the downloading of JPEG images on either a website or through e-mail. If not invalidated, this patent could apply to any website that displays JPEG images. The patent was under reexamination by the U.S. Patent

and Trademark Office from 2000 to 2007; in July 2007, the Patent Office revoked all of the original claims of the patent but found that an additional claim proposed by Global Patent Holdings (claim 17) was valid.[37] Global Patent Holdings then filed a number of lawsuits based on claim 17 of its patent.

In its first two lawsuits following the reexamination, both filed in Chicago, Illinois, Global Patent Holdings sued the Green Bay Packers, CDW, Motorola, Apple, Orbitz, Officemax, Caterpillar, Kraft and Peapod as defendants. A third lawsuit was filed on December 5, 2007, in South Florida against ADT Security Services, AutoNation, Florida Crystals Corp., HearUSA, MovieTickets.com, Ocwen Financial Corp. and Tire Kingdom, and a fourth lawsuit on January 8, 2008, in South Florida against the Boca Raton Resort & Club. A fifth lawsuit was filed against Global Patent Holdings in Nevada. That lawsuit was filed by Zappos.com, Inc., which was allegedly threatened by Global Patent Holdings, and sought a judicial declaration that the '341 patent is invalid and not infringed.

Global Patent Holdings had also used the '341 patent to sue or threaten outspoken critics of broad software patents, including Gregory Aharonian[38] and the anonymous operator of a website blog known as the "Patent Troll Tracker."[39] On December 21, 2007, patent lawyer Vernon Francissen of Chicago asked the U.S. Patent and Trademark Office to reexamine the sole remaining claim of the '341 patent on the basis of new prior art.[40]

On March 5, 2008, the U.S. Patent and Trademark Office agreed to reexamine the '341 patent, finding that the new prior art raised substantial new questions regarding the patent's validity.[41] In light of the reexamination, the accused infringers in four of the five pending lawsuits have filed motions to suspend (stay) their cases until completion of the U.S. Patent and Trademark Office's review of the '341 patent. On April 23, 2008, a judge presiding over the two lawsuits in Chicago, Illinois granted the motions in those cases.[42] On July 22, 2008, the Patent Office issued the first "Office Action" of the second reexamination, finding the claim invalid based on nineteen separate grounds.[43] On Nov. 24, 2009, a Reexamination Certificate was issued cancelling all claims.

Beginning in 2011 and continuing as of early 2013, an entity known as Princeton Digital Image Corporation,[44] based in Eastern Texas, began suing large numbers of companies for alleged infringement of U.S. patent 4,813,056 (https://patents.google.com/patent/US4813056). Princeton claims that the JPEG image compression standard infringes the '056 patent and has sued large numbers of websites, retailers, camera and device manufacturers and resellers. The patent was originally owned and assigned to General Electric. The patent expired in December 2007, but Princeton has sued large numbers of companies for "past infringement" of this patent. (Under U.S. patent laws, a patent owner can sue for "past infringement" up to six years before the filing of a lawsuit, so Princeton could theoretically have continued suing companies until December 2013.) As of March 2013, Princeton had suits pending in New York and Delaware against more than 55 companies. General Electric's involvement in the suit is unknown, although court records indicate that it assigned the patent to Princeton in 2009 and retains certain rights in the patent.[45]

Typical use

The JPEG compression algorithm operates at its best on photographs and paintings of realistic scenes with smooth variations of tone and color. For web usage, where reducing the amount of data used for an image is important for responsive presentation, JPEG's compression benefits make JPEG popular. JPEG/Exif is also the most common format saved by digital cameras.

However, JPEG is not well suited for line drawings and other textual or iconic graphics, where the sharp contrasts between adjacent pixels can cause noticeable artifacts.[46] Such images are better saved in a lossless graphics format such as TIFF, GIF, PNG, or a raw image format. The JPEG standard includes a lossless coding mode, but that mode is not supported in most products.

As the typical use of JPEG is a lossy compression method, which reduces the image fidelity, it is inappropriate for exact reproduction of imaging data (such as some scientific and medical imaging applications and certain technical image processing work).[46]

JPEG is also not well suited to files that will undergo multiple edits, as some image quality is lost each time the image is recompressed, particularly if the image is cropped or shifted, or if encoding parameters are changed – see digital generation loss for details. To prevent image information loss during sequential and repetitive editing, the first edit can be saved in a lossless format, subsequently edited in that format, then finally published as JPEG for distribution.

JPEG compression

JPEG uses a lossy form of compression based on the discrete cosine transform (DCT). This mathematical operation converts each frame/field of the video source from the spatial (2D) domain into the frequency domain (a.k.a. transform domain). A perceptual model based loosely on how the human psychovisual system discards high-frequency information, i.e. sharp transitions in intensity, and color hue. In the transform domain, the process of reducing information is called quantization. In simpler terms, quantization is a method for optimally reducing a large number scale (with different occurrences of each number) into a smaller one, and the transform-domain is a convenient representation of the image because the high-frequency coefficients, which contribute less to the overall picture than other coefficients, are characteristically small-values with high compressibility. The quantized coefficients are then sequenced and losslessly packed into the output bitstream. Nearly all software implementations of JPEG permit user control over the compression ratio (as well as other optional parameters), allowing the user to trade off picture-quality for smaller file size. In embedded applications (such as miniDV, which uses a similar DCT-compression scheme), the parameters are pre-selected and fixed for the application.

The compression method is usually lossy, meaning that some original image information is lost and cannot be restored, possibly affecting image quality. There is an optional lossless mode defined in the JPEG standard. However, this mode is not widely supported in products.

There is also an interlaced *progressive* JPEG format, in which data is compressed in multiple passes of progressively higher detail. This is ideal for large images that will be displayed while downloading over a slow connection, allowing a reasonable preview after receiving only a portion of the data. However, support for progressive JPEGs is not universal. When progressive JPEGs are received by programs that do not support them (such as versions of Internet Explorer before Windows 7)[47] the software displays the image only after it has been completely downloaded.

There are also many medical imaging, traffic and camera applications that create and process 12-bit JPEG images both grayscale and color. 12-bit JPEG format is included in an Extended part of the JPEG specification. The libjpeg codec supports 12-bit JPEG and there even exists a high-performance version.[48]

Lossless editing

Several alterations to a JPEG image can be performed losslessly (that is, without recompression and the associated quality loss) as long as the image size is a multiple of 1 MCU block (Minimum Coded Unit) (usually 16 pixels in both directions, for 4:2:0 chroma subsampling). Utilities that implement this include:

- jpegtran and its GUI, Jpegcrop.
- IrfanView using "JPG Lossless Crop (PlugIn)" and "JPG Lossless Rotation (PlugIn)", which require installing the JPG_TRANSFORM plugin.
- FastStone Image Viewer using "Lossless Crop to File" and "JPEG Lossless Rotate".
- XnViewMP using "JPEG lossless transformations".
- ACDSee supports lossless rotation (but not lossless cropping) with its "Force lossless JPEG operations" option.

Blocks can be rotated in 90-degree increments, flipped in the horizontal, vertical and diagonal axes and moved about in the image. Not all blocks from the original image need to be used in the modified one.

The top and left edge of a JPEG image must lie on an 8 × 8 pixel block boundary (or 16 × 16 pixel for larger MCU sizes), but the bottom and right edge need not do so. This limits the possible **lossless crop** operations, and prevents flips and rotations of an image whose bottom or right edge does not lie on a block boundary for all channels (because the edge would end up on top or left, where – as aforementioned – a block boundary is obligatory).

Rotations where the image is not a multiple of 8 or 16, which value depends upon the chroma subsampling, are not lossless. Rotating such an image causes the blocks to be recomputed which results in loss of quality.[49]

When using lossless cropping, if the bottom or right side of the crop region is not on a block boundary, then the rest of the data from the partially used blocks will still be present in the cropped file and can be recovered. It is also possible to transform between baseline and progressive formats without any loss of quality, since the only difference is the order in which the coefficients are placed in the file.

Furthermore, several JPEG images can be losslessly joined, as long as they were saved with the same quality and the edges coincide with block boundaries.

JPEG files

The file format known as "JPEG Interchange Format" (JIF) is specified in Annex B of the standard. However, this "pure" file format is rarely used, primarily because of the difficulty of programming encoders and decoders that fully implement all aspects of the standard and because of certain shortcomings of the standard:

- Color space definition
- Component sub-sampling registration
- Pixel aspect ratio definition.

Several additional standards have evolved to address these issues. The first of these, released in 1992, was the JPEG File Interchange Format (JFIF), followed in recent years by Exchangeable image file format (Exif) and ICC color profiles. Both of these formats use the actual JIF byte layout, consisting of

different *markers*, but in addition, employ one of the JIF standard's extension points, namely the *application markers*: JFIF uses APP0, while Exif uses APP1. Within these segments of the file that were left for future use in the JIF standard and are not read by it, these standards add specific metadata.

Thus, in some ways, JFIF is a cut-down version of the JIF standard in that it specifies certain constraints (such as not allowing all the different encoding modes), while in other ways, it is an extension of JIF due to the added metadata. The documentation for the original JFIF standard states:[50]

> JPEG File Interchange Format is a minimal file format which enables JPEG bitstreams to be exchanged between a wide variety of platforms and applications. This minimal format does not include any of the advanced features found in the TIFF JPEG specification or any application specific file format. Nor should it, for the only purpose of this simplified format is to allow the exchange of JPEG compressed images.

Image files that employ JPEG compression are commonly called "JPEG files", and are stored in variants of the JIF image format. Most image capture devices (such as digital cameras) that output JPEG are actually creating files in the Exif format, the format that the camera industry has standardized on for metadata interchange. On the other hand, since the Exif standard does not allow color profiles, most image editing software stores JPEG in JFIF format, and includes the APP1 segment from the Exif file to include the metadata in an almost-compliant way; the JFIF standard is interpreted somewhat flexibly.[51]

Strictly speaking, the JFIF and Exif standards are incompatible, because each specifies that its marker segment (APP0 or APP1, respectively) appear first. In practice, most JPEG files contain a JFIF marker segment that precedes the Exif header. This allows older readers to correctly handle the older format JFIF segment, while newer readers also decode the following Exif segment, being less strict about requiring it to appear first.

JPEG filename extensions

The most common filename extensions for files employing JPEG compression are **`.jpg`** and **`.jpeg`**, though `.jpe`, `.jfif` and `.jif` are also used.[52] It is also possible for JPEG data to be embedded in other file types – TIFF encoded files often embed a JPEG image as a thumbnail of the main image; and MP3 files can contain a JPEG of cover art in the ID3v2 tag.

Color profile

Many JPEG files embed an ICC color profile (color space). Commonly used color profiles include sRGB and Adobe RGB. Because these color spaces use a non-linear transformation, the dynamic range of an 8-bit JPEG file is about 11 stops; see gamma curve.

If the image doesn't specify color profile information (*untagged*), the color space is assumed to be sRGB for the purposes of display on webpages.[53][54]

Syntax and structure

A JPEG image consists of a sequence of *segments*, each beginning with a *marker*, each of which begins with a 0xFF byte, followed by a byte indicating what kind of marker it is. Some markers consist of just those two bytes; others are followed by two bytes (high then low), indicating the length of marker-specific payload data that follows. (The length includes the two bytes for the length, but not the two bytes for the marker.) Some markers are followed by entropy-coded data; the length of such a marker does not include the entropy-coded data. Note that consecutive 0xFF bytes are used as fill bytes for padding purposes, although this fill byte padding should only ever take place for markers immediately following entropy-coded scan data (see JPEG specification section B.1.1.2 and E.1.2 for details; specifically "In all cases where markers are appended after the compressed data, optional 0xFF fill bytes may precede the marker").

Within the entropy-coded data, after any 0xFF byte, a 0x00 byte is inserted by the encoder before the next byte, so that there does not appear to be a marker where none is intended, preventing framing errors. Decoders must skip this 0x00 byte. This technique, called byte stuffing (see JPEG specification section F.1.2.3), is only applied to the entropy-coded data, not to marker payload data. Note however that entropy-coded data has a few markers of its own; specifically the Reset markers (0xD0 through 0xD7), which are used to isolate independent chunks of entropy-coded data to allow parallel decoding, and encoders are free to insert these Reset markers at regular intervals (although not all encoders do this).

Common JPEG markers[55]

Short name	Bytes	Payload	Name	Comments
SOI	0xFF, 0xD8	*none*	Start Of Image	
SOF0	0xFF, 0xC0	*variable size*	Start Of Frame (baseline DCT)	Indicates that this is a baseline DCT-based JPEG, and specifies the width, height, number of components, and component subsampling (e.g., 4:2:0).
SOF2	0xFF, 0xC2	*variable size*	Start Of Frame (progressive DCT)	Indicates that this is a progressive DCT-based JPEG, and specifies the width, height, number of components, and component subsampling (e.g., 4:2:0).
DHT	0xFF, 0xC4	*variable size*	Define Huffman Table(s)	Specifies one or more Huffman tables.
DQT	0xFF, 0xDB	*variable size*	Define Quantization Table(s)	Specifies one or more quantization tables.
DRI	0xFF, 0xDD	4 bytes	Define Restart Interval	Specifies the interval between RST*n* markers, in Minimum Coded Units (MCUs). This marker is followed by two bytes indicating the fixed size so it can be treated like any other variable size segment.
SOS	0xFF, 0xDA	*variable size*	Start Of Scan	Begins a top-to-bottom scan of the image. In baseline DCT JPEG images, there is generally a single scan. Progressive DCT JPEG images usually contain multiple scans. This marker specifies which slice of data it will contain, and is immediately followed by entropy-coded data.
RST*n*	0xFF, 0xD*n* (*n*=0..7)	*none*	Restart	Inserted every *r* macroblocks, where *r* is the restart interval set by a DRI marker. Not used if there was no DRI marker. The low three bits of the marker code cycle in value from 0 to 7.
APP*n*	0xFF, 0xE*n*	*variable size*	Application-specific	For example, an Exif JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely on TIFF.
COM	0xFF, 0xFE	*variable size*	Comment	Contains a text comment.
EOI	0xFF, 0xD9	*none*	End Of Image	

There are other *Start Of Frame* markers that introduce other kinds of JPEG encodings.

Since several vendors might use the same APP*n* marker type, application-specific markers often begin with a standard or vendor name (e.g., "Exif" or "Adobe") or some other identifying string.

At a restart marker, block-to-block predictor variables are reset, and the bitstream is synchronized to a byte boundary. Restart markers provide means for recovery after bitstream error, such as transmission over an unreliable network or file corruption. Since the runs of macroblocks between restart markers may be independently decoded, these runs may be decoded in parallel.

JPEG codec example

Although a JPEG file can be encoded in various ways, most commonly it is done with JFIF encoding. The encoding process consists of several steps:

1. The representation of the colors in the image is converted from RGB to $Y'C_BC_R$, consisting of one luma component (Y'), representing brightness, and two chroma components, (C_B and C_R), representing color. This step is sometimes skipped.
2. The resolution of the chroma data is reduced, usually by a factor of 2 or 3. This reflects the fact that the eye is less sensitive to fine color details than to fine brightness details.
3. The image is split into blocks of 8×8 pixels, and for each block, each of the Y, C_B, and C_R data undergoes the discrete cosine transform (DCT). A DCT is similar to a Fourier transform in the sense that it produces a kind of spatial frequency spectrum.
4. The amplitudes of the frequency components are quantized. Human vision is much more sensitive to small variations in color or brightness over large areas than to the strength of high-frequency brightness variations. Therefore, the magnitudes of the high-frequency components are stored with a lower accuracy than the low-frequency components. The quality setting of the encoder (for example 50 or 95 on a scale of 0–100 in the Independent JPEG Group's library[56]) affects to what extent the resolution of each frequency component is reduced. If an excessively low quality setting is used, the high-frequency components are discarded altogether.
5. The resulting data for all 8×8 blocks is further compressed with a lossless algorithm, a variant of Huffman encoding.

The decoding process reverses these steps, except the *quantization* because it is irreversible. In the remainder of this section, the encoding and decoding processes are described in more detail.

Encoding

Many of the options in the JPEG standard are not commonly used, and as mentioned above, most image software uses the simpler JFIF format when creating a JPEG file, which among other things specifies the encoding method. Here is a brief description of one of the more common methods of encoding when applied to an input that has 24 bits per pixel (eight each of red, green, and blue). This particular option is a lossy data compression method. They are represented in matrices below.

Color space transformation

First, the image should be converted from RGB (by default sRGB,[53][54] but other color spaces are possible) into a different color space called $Y'C_BC_R$ (or, informally, YCbCr). It has three components Y', C_B and C_R: the Y' component represents the brightness of a pixel, and the C_B and C_R components represent the chrominance (split into blue and red components). This is basically the same color space as used by digital color television as well as digital video including video DVDs. The $Y'C_BC_R$ color space conversion allows greater compression without a significant effect on perceptual image quality (or greater perceptual image quality for the same compression). The compression is more efficient because the brightness information, which is more important to the eventual perceptual quality of the image, is confined to a single channel. This more closely corresponds to the perception of color in the human visual system. The color transformation also improves compression by statistical decorrelation.

A particular conversion to $Y'C_BC_R$ is specified in the JFIF standard, and should be performed for the resulting JPEG file to have maximum compatibility. However, some JPEG implementations in "highest quality" mode do not apply this step and instead keep the color information in the RGB color model,[57] where the image is stored in separate channels for red, green and blue brightness components. This results in less efficient compression, and would not likely be used when file size is especially important.

Downsampling

Due to the densities of color- and brightness-sensitive receptors in the human eye, humans can see considerably more fine detail in the brightness of an image (the Y' component) than in the hue and color saturation of an image (the Cb and Cr components). Using this knowledge, encoders can be designed to compress images more efficiently.

The transformation into the $Y'C_BC_R$ color model enables the next usual step, which is to reduce the spatial resolution of the Cb and Cr components (called "downsampling" or "chroma subsampling"). The ratios at which the downsampling is ordinarily done for JPEG images are 4:4:4 (no downsampling), 4:2:2 (reduction by a factor of 2 in the horizontal direction), or (most commonly) 4:2:0 (reduction by a factor of 2 in both the horizontal and vertical directions). For the rest of the compression process, Y', Cb and Cr are processed separately and in a very similar manner.

Block splitting

After subsampling, each channel must be split into 8×8 blocks. Depending on chroma subsampling, this yields Minimum Coded Unit (MCU) blocks of size 8×8 (4:4:4 – no subsampling), 16×8 (4:2:2), or most commonly 16×16 (4:2:0). In video compression MCUs are called macroblocks.

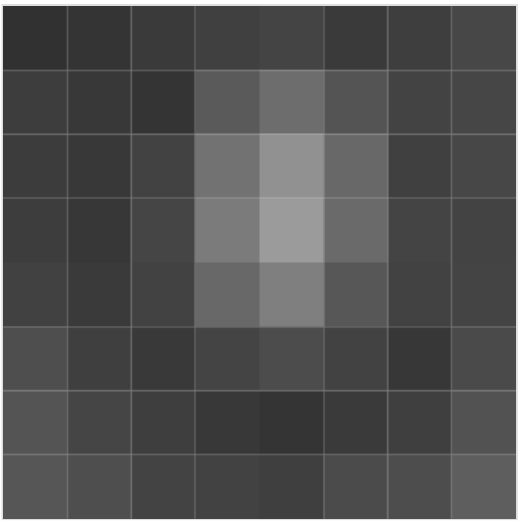
If the data for a channel does not represent an integer number of blocks then the encoder must fill the remaining area of the incomplete blocks with some form of dummy data. Filling the edges with a fixed color (for example, black) can create ringing artifacts along the visible part of the border; repeating the edge pixels is a common technique that reduces (but does not necessarily eliminate) such artifacts, and more sophisticated border filling techniques can also be applied.

Discrete cosine transform

Next, each 8×8 block of each component (Y, Cb, Cr) is converted to a frequency-domain representation, using a normalized, two-dimensional type-II discrete cosine transform (DCT), see Citation 1 in discrete cosine transform. The DCT is sometimes referred to as "type-II DCT" in the context of a family of transforms as in discrete cosine transform, and the corresponding inverse (IDCT) is denoted as "type-III DCT".

As an example, one such 8×8 8-bit subimage might be:

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}.$$



The 8×8 sub-image shown in 8-bit grayscale

Before computing the DCT of the 8×8 block, its values are shifted from a positive range to one centered on zero. For an 8-bit image, each entry in the original block falls in the range $[0, 255]$. The midpoint of the range (in this case, the value 128) is subtracted from each entry to produce a data range that is centered on zero, so that the modified range is $[-128, 127]$. This step reduces the dynamic range requirements in the DCT processing stage that follows.

This step results in the following values:

$$g = \begin{array}{cc} & \begin{array}{c} x \\ \longrightarrow \end{array} \\ & \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix} \end{array} \Big\downarrow y.$$

The next step is to take the two-dimensional DCT, which is given by:

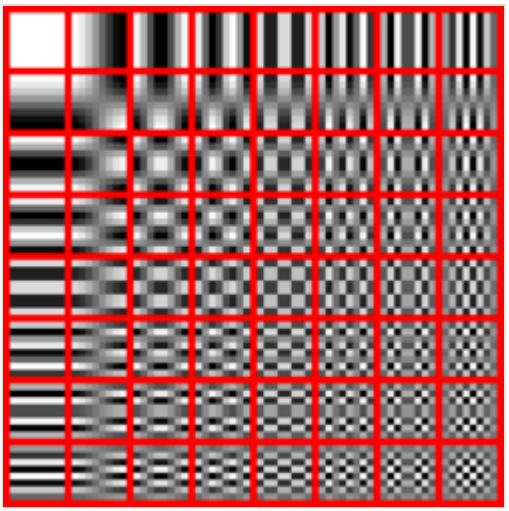
$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

where

- u is the horizontal spatial frequency, for the integers $0 \le u < 8$.
- v is the vertical spatial frequency, for the integers $0 \le v < 8$.
- $\alpha(u)$ and $\alpha(v)$ are normalizing scale factors to make the transformation orthonormal with $\alpha(i) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } i = 0 \\ 1, & \text{otherwise} \end{cases}$
- $g_{x,y}$ is the pixel value at coordinates (x, y)

- $G_{u,v}$ is the DCT coefficient at coordinates (u, v).

If we perform this transformation on our matrix above, we get the following (rounded to the nearest two digits beyond the decimal point):



The DCT transforms an 8×8 block of input values to a linear combination of these 64 patterns. The patterns are referred to as the two-dimensional DCT *basis functions*, and the output values are referred to as *transform coefficients*. The horizontal index is u and the vertical index is v.

$$G = \begin{array}{c} \xrightarrow{\quad u \quad} \\ \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \end{array} \begin{array}{c} \\ \\ \\ \downarrow v. \\ \\ \\ \\ \\ \end{array}$$

Note the top-left corner entry with the rather large magnitude. This is the DC coefficient (also called the constant component), which defines the basic hue for the entire block. The remaining 63 coefficients are the AC coefficients (also called the alternating components).[58] The advantage of the DCT is its tendency to aggregate most of the signal in one corner of the result, as may be seen above. The quantization step to follow accentuates this effect while simultaneously reducing the overall size of the DCT coefficients, resulting in a signal that is easy to compress efficiently in the entropy stage.

The DCT temporarily increases the bit-depth of the data, since the DCT coefficients of an 8-bit/component image take up to 11 or more bits (depending on fidelity of the DCT calculation) to store. This may force the codec to temporarily use 16-bit numbers to hold these coefficients, doubling the size of the image representation at this point; these values are typically reduced back to 8-bit values by the quantization step. The temporary increase in size at this stage is not a performance concern for most JPEG implementations, since typically only a very small part of the image is stored in full DCT form at any given time during the image encoding or decoding process.

Quantization

The human eye is good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. This allows one to greatly reduce the amount of information in the high frequency components. This is done by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer. This rounding operation is the only lossy operation in the whole process (other than chroma subsampling) if the DCT computation is performed with sufficiently high precision. As a result of this, it is typically the case that many of the higher frequency components are rounded to zero, and many of the rest become small positive or negative numbers, which take many fewer bits to represent.

The elements in the quantization matrix control the compression ratio, with larger values producing greater compression. A typical quantization matrix (for a quality of 50% as specified in the original JPEG Standard), is as follows:

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

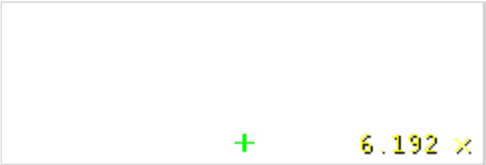
The quantized DCT coefficients are computed with

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0, 1, 2, \ldots, 7; k = 0, 1, 2, \ldots, 7$$

where G is the unquantized DCT coefficients; Q is the quantization matrix above; and B is the quantized DCT coefficients.

Using this quantization matrix with the DCT coefficient matrix from above results in:

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$



Left: a final image is built up from a series of basis functions. Right: each of the DCT basis functions that comprise the image, and the corresponding weighting coefficient. Middle: the basis function, after multiplication by the coefficient: this component is added to the final image. For clarity, the 8×8 macroblock in this example is magnified by 10x using bilinear interpolation.

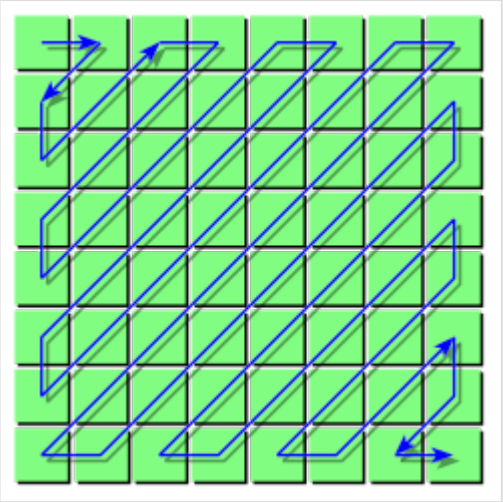
For example, using −415 (the DC coefficient) and rounding to the nearest integer

$$\text{round}\left(\frac{-415.37}{16}\right) = \text{round}\left(-25.96\right) = -26.$$

Notice that most of the higher-frequency elements of the sub-block (i.e., those with an *x* or *y* spatial frequency greater than 4) are quantized into zero values.

Entropy coding

Entropy coding is a special form of lossless data compression. It involves arranging the image components in a "zigzag" order employing run-length encoding (RLE) algorithm that groups similar frequencies together, inserting length coding zeros, and then using Huffman coding on what is left.



Zigzag ordering of JPEG image components

The JPEG standard also allows, but does not require, decoders to support the use of arithmetic coding, which is mathematically superior to Huffman coding. However, this feature has rarely been used, as it was historically covered by patents requiring royalty-bearing licenses, and because it is slower to encode and decode compared to Huffman coding. Arithmetic coding typically makes files about 5–7% smaller.[59]

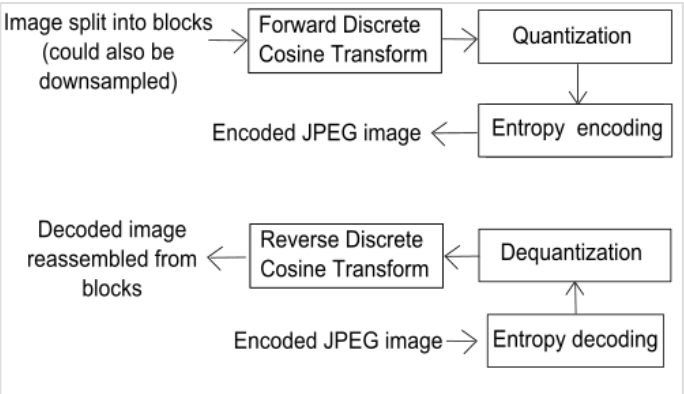
The previous quantized DC coefficient is used to predict the current quantized DC coefficient. The difference between the two is encoded rather than the actual value. The encoding of the 63 quantized AC coefficients does not use such prediction differencing.

The zigzag sequence for the above quantized coefficients are shown below. (The format shown is just for ease of understanding/viewing.)

−26							
−3	0						
−3	−2	−6					
2	−4	1	−3				
1	1	5	1	2			
−1	1	−1	2	0	0		
0	0	0	−1	−1	0	0	
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
0	0	0	0	0	0		
0	0	0	0	0			
0	0	0	0				
0	0	0					
0	0						
0							

If the *i*-th block is represented by B_i and positions within each block are represented by (p, q) where $p = 0, 1, \ldots, 7$ and $q = 0, 1, \ldots, 7$, then any coefficient in the DCT image can be represented as $B_i(p, q)$. Thus, in the above scheme, the order of encoding pixels (for the i-th block) is $B_i(0, 0)$, $B_i(0, 1)$, $B_i(1, 0)$, $B_i(2, 0)$, $B_i(1, 1)$, $B_i(0, 2)$, $B_i(0, 3)$, $B_i(1, 2)$ and so on.

This encoding mode is called baseline *sequential* encoding. Baseline JPEG also supports *progressive* encoding. While sequential encoding encodes coefficients of a single block at a time (in a zigzag manner), progressive encoding encodes similar-positioned batch of coefficients of all blocks in one go (called a *scan*), followed by the next batch of coefficients of all blocks, and so on. For example, if the image is divided into N 8×8 blocks $B_0, B_1, B_2, \ldots, B_{n-1}$, then a 3-scan progressive encoding encodes DC component, $B_i(0,0)$ for all blocks, i.e., for all $i = 0, 1, 2, \ldots, N-1$, in first scan. This is followed by the second scan which encoding a few more components (assuming four more components, they are $B_i(0,1)$ to $B_i(1,1)$, still in a zigzag manner) coefficients of all blocks (so the sequence is: $B_0(0,1), B_0(1,0), B_0(2,0), B_0(1,1), B_1(0,1), B_1(1,0), \ldots, B_N(2,0), B_N(1,1)$), followed by all the remained coefficients of all blocks in the last scan.



Baseline sequential JPEG encoding and decoding processes

Once all similar-positioned coefficients have been encoded, the next position to be encoded is the one occurring next in the zigzag traversal as indicated in the figure above. It has been found that *baseline progressive* JPEG encoding usually gives better compression as compared to *baseline sequential* JPEG due to the ability to use different Huffman tables (see below) tailored for different frequencies on each "scan" or "pass" (which includes similar-positioned coefficients), though the difference is not too large.

In the rest of the article, it is assumed that the coefficient pattern generated is due to sequential mode.

In order to encode the above generated coefficient pattern, JPEG uses Huffman encoding. The JPEG standard provides general-purpose Huffman tables; encoders may also choose to generate Huffman tables optimized for the actual frequency distributions in images being encoded.

The process of encoding the zig-zag quantized data begins with a run-length encoding explained below, where:

- *x* is the non-zero, quantized AC coefficient.
- *RUNLENGTH* is the number of zeroes that came before this non-zero AC coefficient.
- *SIZE* is the number of bits required to represent *x*.
- *AMPLITUDE* is the bit-representation of *x*.

The run-length encoding works by examining each non-zero AC coefficient *x* and determining how many zeroes came before the previous AC coefficient. With this information, two symbols are created:

Symbol 1	Symbol 2
(RUNLENGTH, SIZE)	(AMPLITUDE)

Both *RUNLENGTH* and *SIZE* rest on the same byte, meaning that each only contains four bits of information. The higher bits deal with the number of zeroes, while the lower bits denote the number of bits necessary to encode the value of *x*.

This has the immediate implication of *Symbol 1* being only able store information regarding the first 15 zeroes preceding the non-zero AC coefficient. However, JPEG defines two special Huffman code words. One is for ending the sequence prematurely when the remaining coefficients are zero (called "End-of-Block" or "EOB"), and another when the run of zeroes goes beyond 15 before reaching a non-zero AC coefficient. In such a case where 16 zeroes are encountered before a given non-zero AC coefficient, *Symbol 1* is encoded "specially" as: (15, 0)(0).

The overall process continues until "EOB" – denoted by (0, 0) – is reached.

With this in mind, the sequence from earlier becomes:

> (0, 2)(-3);(1, 2)(-3);(0, 2)(-2);(0, 3)(-6);(0, 2)(2);(0, 3)(-4);(0, 1)(1);(0, 2)(-3);(0, 1)(1);(0, 1)(1);
> (0, 3)(5);(0, 1)(1);(0, 2)(2);(0, 1)(-1);(0, 1)(1);(0, 1)(-1);(0, 2)(2);(5, 1)(-1);(0, 1)(-1);(0, 0);

(The first value in the matrix, −26, is the DC coefficient; it is not encoded the same way. See above.)

From here, frequency calculations are made based on occurrences of the coefficients. In our example block, most of the quantized coefficients are small numbers that are not preceded immediately by a zero coefficient. These more-frequent cases will be represented by shorter code words.

Compression ratio and artifacts

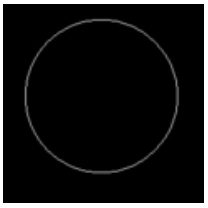
The resulting compression ratio can be varied according to need by being more or less aggressive in the divisors used in the quantization phase. Ten to one compression usually results in an image that cannot be distinguished by eye from the original. A compression ratio of 100:1 is usually possible, but will look distinctly artifacted compared to the original. The appropriate level of compression depends on the use to which the image will be put.

This image shows the pixels that are different between a non-compressed image and the same image JPEG compressed with a quality setting of 50. Darker means a larger difference. Note especially the changes occurring near sharp edges and having a block-like shape.

Those who use the World Wide Web may be familiar with the irregularities known as compression artifacts that appear in JPEG images, which may take the form of noise around contrasting edges (especially curves and corners), or "blocky" images. These are due to the quantization step of the JPEG algorithm. They are especially noticeable around sharp corners between contrasting colors (text is a good example, as it contains many such corners). The analogous artifacts in MPEG video are referred to as *mosquito noise*, as the resulting "edge busyness" and spurious dots, which change over time, resemble mosquitoes swarming around the object.[60][61]

These artifacts can be reduced by choosing a lower level of compression; they may be completely avoided by saving an image using a lossless file format, though this will result in a larger file size. The images created with ray-tracing programs have noticeable blocky shapes on the terrain. Certain low-

intensity compression artifacts might be acceptable when simply viewing the images, but can be emphasized if the image is subsequently processed, usually resulting in unacceptable quality. Consider the example below, demonstrating the effect of lossy compression on an edge detection processing step.

Image	Lossless compression	Lossy compression
Original		
Processed by Canny edge detector		

The original image

The compressed 8×8 squares are visible in the scaled-up picture, together with other visual artifacts of the lossy compression.

Some programs allow the user to vary the amount by which individual blocks are compressed. Stronger compression is applied to areas of the image that show fewer artifacts. This way it is possible to manually reduce JPEG file size with less loss of quality.

Since the quantization stage *always* results in a loss of information, JPEG standard is always a lossy compression codec. (Information is lost both in quantizing and rounding of the floating-point numbers.) Even if the quantization matrix is a matrix of ones, information will still be lost in the rounding step.

External image

Illustration of edge busyness (http://i.cmpnet.com/videsignline/2006/02/algolith-fig2.jpg)[60]

Decoding

Decoding to display the image consists of doing all the above in reverse.

Taking the DCT coefficient matrix (after adding the difference of the DC coefficient back in)

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and taking the entry-for-entry product with the quantization matrix from above results in

$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

which closely resembles the original DCT coefficient matrix for the top-left portion.

The next step is to take the two-dimensional inverse DCT (a 2D type-III DCT), which is given by:

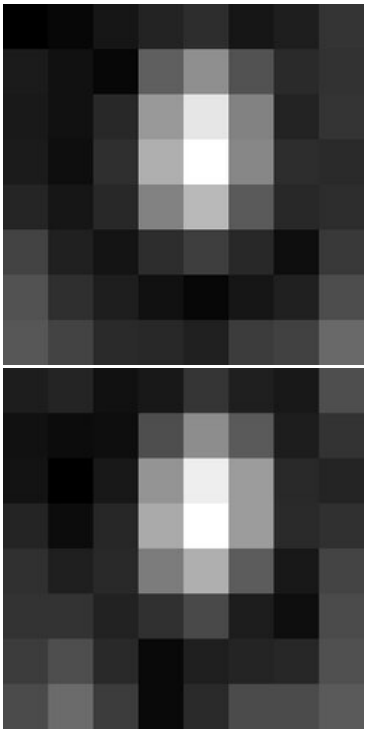
$$f_{x,y} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} \alpha(u)\alpha(v) F_{u,v} \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

where

- x is the pixel row, for the integers $0 \le x < 8$.
- y is the pixel column, for the integers $0 \le y < 8$.
- $\alpha(u)$ is the normalizing scale factor defined earlier, for the integers $0 \le u < 8$.
- $F_{u,v}$ is the approximated DCT coefficient at coordinates (u, v).
- $f_{x,y}$ is the reconstructed pixel value at coordinates (x, y)

Rounding the output to integer values (since the original had integer values) results in an image with values (still shifted down by 128)

$$\begin{bmatrix} -66 & -63 & -71 & -68 & -56 & -65 & -68 & -46 \\ -71 & -73 & -72 & -46 & -20 & -41 & -66 & -57 \\ -70 & -78 & -68 & -17 & 20 & -14 & -61 & -63 \\ -63 & -73 & -62 & -8 & 27 & -14 & -60 & -58 \\ -58 & -65 & -61 & -27 & -6 & -40 & -68 & -50 \\ -57 & -57 & -64 & -58 & -48 & -66 & -72 & -47 \\ -53 & -46 & -61 & -74 & -65 & -63 & -62 & -45 \\ -47 & -34 & -53 & -74 & -60 & -47 & -47 & -41 \end{bmatrix}$$



Slight differences are noticeable between the original (top) and decompressed image (bottom), which is most readily seen in the bottom-left corner.

and adding 128 to each entry

$$\begin{bmatrix} 62 & 65 & 57 & 60 & 72 & 63 & 60 & 82 \\ 57 & 55 & 56 & 82 & 108 & 87 & 62 & 71 \\ 58 & 50 & 60 & 111 & 148 & 114 & 67 & 65 \\ 65 & 55 & 66 & 120 & 155 & 114 & 68 & 70 \\ 70 & 63 & 67 & 101 & 122 & 88 & 60 & 78 \\ 71 & 71 & 64 & 70 & 80 & 62 & 56 & 81 \\ 75 & 82 & 67 & 54 & 63 & 65 & 66 & 83 \\ 81 & 94 & 75 & 54 & 68 & 81 & 81 & 87 \end{bmatrix}.$$

This is the decompressed subimage. In general, the decompression process may produce values outside the original input range of $[0, 255]$. If this occurs, the decoder needs to clip the output values so as to keep them within that range to prevent overflow when storing the decompressed image with the original bit depth.

The decompressed subimage can be compared to the original subimage (also see images to the right) by taking the difference (original − uncompressed) results in the following error values:

$$\begin{bmatrix} -10 & -10 & 4 & 6 & -2 & -2 & 4 & -9 \\ 6 & 4 & -1 & 8 & 1 & -2 & 7 & 1 \\ 4 & 9 & 8 & 2 & -4 & -10 & -1 & 8 \\ -2 & 3 & 5 & 2 & -1 & -8 & 2 & -1 \\ -3 & -2 & 1 & 3 & 4 & 0 & 8 & -8 \\ 8 & -6 & -4 & -0 & -3 & 6 & 2 & -6 \\ 10 & -11 & -3 & 5 & -8 & -4 & -1 & -0 \\ 6 & -15 & -6 & 14 & -3 & -5 & -3 & 7 \end{bmatrix}$$

with an average absolute error of about 5 values per pixels (i.e., $\frac{1}{64}\sum_{x=0}^{7}\sum_{y=0}^{7}|e(x,y)| = 4.8750$).

The error is most noticeable in the bottom-left corner where the bottom-left pixel becomes darker than the pixel to its immediate right.

Required precision

The required implementation precision of a JPEG codec is implicitly defined through the requirements formulated for compliance to the JPEG standard. These requirements are specified in ITU.T Recommendation T.83 | ISO/IEC 10918-2. Unlike MPEG standards and many later JPEG standards, the above document defines both required implementation precisions for the encoding and the decoding process of a JPEG codec by means of a maximal tolerable error of the forwards and inverse DCT in the DCT domain as determined by reference test streams. For example, the output of a decoder implementation must not exceed an error of one quantization unit in the DCT domain when applied to the reference testing codestreams provided as part of the above standard. While unusual, and unlike many other and more modern standards, ITU.T T.83 | ISO/IEC 10918-2 does not formulate error bounds in the image domain.

Effects of JPEG compression

JPEG compression artifacts blend well into photographs with detailed non-uniform textures, allowing higher compression ratios. Notice how a higher compression ratio first affects the high-frequency textures in the upper-left corner of the image, and how the contrasting lines become more fuzzy. The very high compression ratio severely affects the quality of the image, although the overall colors and image form are still recognizable. However, the precision of colors suffer less (for a human eye) than the precision of contours (based on luminance). This justifies the fact that images should be first transformed in a color model separating the luminance from the chromatic information, before subsampling the chromatic planes (which may also use lower quality quantization) in order to preserve the precision of the luminance plane with more information bits.

Sample photographs

For information, the uncompressed 24-bit RGB bitmap image below (73,242 pixels) would require 219,726 bytes (excluding all other information headers). The filesizes indicated below include the internal JPEG information headers and some metadata. For highest quality images (Q=100), about 8.25

bits per color pixel is required. On grayscale images, a minimum of 6.5 bits per pixel is enough (a comparable Q=100 quality color information requires about 25% more encoded bits). The highest quality image below (Q=100) is encoded at nine bits per color pixel, the medium quality image (Q=25) uses one bit per color pixel. For most applications, the quality factor should not go below 0.75 bit per pixel (Q=12.5), as demonstrated by the low quality image. The image at lowest quality uses only 0.13 bit per pixel, and displays very poor color. This is useful when the image will be displayed in a significantly scaled-down size. A method for creating better quantization matrices for a given image quality using PSNR instead of the Q factor is described in Minguillón & Pujol (2001).[62]

Visual impact of a jpeg compression on Photoshop on a picture of 4480x4480 pixels

Image	Quality	Size (bytes)	Compression ratio	Comment
	Highest quality (Q = 100)	81,447	2.7:1	Extremely minor artifacts
	High quality (Q = 50)	14,679	15:1	Initial signs of subimage artifacts
	Medium quality (Q = 25)	9,407	23:1	Stronger artifacts; loss of high frequency information
	Low quality (Q = 10)	4,787	46:1	Severe high frequency loss leads to obvious artifacts on subimage boundaries ("macroblocking")
	Lowest quality (Q = 1)	1,523	144:1	Extreme loss of color and detail; the leaves are nearly unrecognizable.

Note: The above images are not IEEE / CCIR / EBU test images, and the encoder settings are not specified or available.

The medium quality photo uses only 4.3% of the storage space required for the uncompressed image, but has little noticeable loss of detail or visible artifacts. However, once a certain threshold of compression is passed, compressed images show increasingly visible defects. See the article on rate–distortion theory for

a mathematical explanation of this threshold effect. A particular limitation of JPEG in this regard is its non-overlapped 8×8 block transform structure. More modern designs such as JPEG 2000 and JPEG XR exhibit a more graceful degradation of quality as the bit usage decreases – by using transforms with a larger spatial extent for the lower frequency coefficients and by using overlapping transform basis functions.

Lossless further compression

From 2004 to 2008, new research emerged on ways to further compress the data contained in JPEG images without modifying the represented image.[63][64][65][66] This has applications in scenarios where the original image is only available in JPEG format, and its size needs to be reduced for archiving or transmission. Standard general-purpose compression tools cannot significantly compress JPEG files.

Typically, such schemes take advantage of improvements to the naive scheme for coding DCT coefficients, which fails to take into account:

- Correlations between magnitudes of adjacent coefficients in the same block;
- Correlations between magnitudes of the same coefficient in adjacent blocks;
- Correlations between magnitudes of the same coefficient/block in different channels;
- The DC coefficients when taken together resemble a downscale version of the original image multiplied by a scaling factor. Well-known schemes for lossless coding of continuous-tone images can be applied, achieving somewhat better compression than the Huffman coded DPCM used in JPEG.

Some standard but rarely used options already exist in JPEG to improve the efficiency of coding DCT coefficients: the arithmetic coding option, and the progressive coding option (which produces lower bitrates because values for each coefficient are coded independently, and each coefficient has a significantly different distribution). Modern methods have improved on these techniques by reordering coefficients to group coefficients of larger magnitude together;[63] using adjacent coefficients and blocks to predict new coefficient values;[65] dividing blocks or coefficients up among a small number of independently coded models based on their statistics and adjacent values;[64][65] and most recently, by decoding blocks, predicting subsequent blocks in the spatial domain, and then encoding these to generate predictions for DCT coefficients.[66]

Typically, such methods can compress existing JPEG files between 15 and 25 percent, and for JPEGs compressed at low-quality settings, can produce improvements of up to 65%.[65][66]

A freely available tool called packJPG is based on the 2007 paper "Improved Redundancy Reduction for JPEG Files." As of version 2.5k of 2016, it reports a typical 20% reduction by transcoding.[67] JPEG XL (ISO/IEC 18181) of 2018 reports a similar reduction in its transcoding.

Derived formats for stereoscopic 3D

JPEG Stereoscopic

An example of a stereoscopic .JPS file

JPS is a stereoscopic JPEG image used for creating 3D effects from 2D images. It contains two static images, one for the left eye and one for the right eye; encoded as two side-by-side images in a single JPG file. JPEG Stereoscopic (JPS, extension .jps) is a JPEG-based format for stereoscopic images.[68][69] It has a range of configurations stored in the JPEG APP3 marker field, but usually contains one image of double width, representing two images of identical size in cross-eyed (i.e. left frame on the right half of the image and vice versa) side-by-side arrangement. This file format can be viewed as a JPEG without any special software, or can be processed for rendering in other modes.

JPEG Multi-Picture Format

JPEG Multi-Picture

Filename extension	`.mpo`
Uniform Type Identifier (UTI)	public.mpo-image[70]

JPEG Multi-Picture Format (MPO, extension .mpo) is a JPEG-based format for storing multiple images in a single file. It contains two or more JPEG files concatenated together.[71][72] It also defines a JPEG APP2 marker segment for image description. Various devices use it to store 3D images, such as Fujifilm FinePix Real 3D W1, HTC Evo 3D, JVC GY-HMZ1U AVCHD/MVC extension camcorder, Nintendo 3DS, Panasonic Lumix DMC-TZ20, DMC-TZ30, DMC-TZ60, DMC-TS4 (FT4), and Sony DSC-HX7V. Other devices use it to store "preview images" that can be displayed on a TV.

In the last few years, due to the growing use of stereoscopic images, much effort has been spent by the scientific community to develop algorithms for stereoscopic image compression.[73][74]

Implementations

A very important implementation of a JPEG codec is the free programming library **libjpeg** of the Independent JPEG Group. It was first published in 1991 and was key for the success of the standard. This library was used in countless applications.[3] The development went quiet in 1998; when libjpeg resurfaced with the 2009 version 7, it broke ABI compatibility with previous versions. Version 8 of 2010 introduced non-standard extensions, a decision criticized by the original IJG leader Tom Lane.[75]

libjpeg-turbo, forked from the 1998 libjpeg 6b, improves on libjpeg with SIMD optimizations. Originally seen as a maintained fork of libjpeg, it has become more popular after the incompatible changes of 2009.[76][77] In 2019, it became the ITU|ISO/IEC reference implementation as ISO/IEC 10918-7 and ITU-T T.873.[78]

ISO/IEC Joint Photography Experts Group maintains the other reference software implementation under the JPEG XT heading. It can encode both base JPEG (ISO/IEC 10918-1 and 18477–1) and JPEG XT extensions (ISO/IEC 18477 Parts 2 and 6–9), as well as JPEG-LS (ISO/IEC 14495).[79] In 2016, "JPEG on steroids" was introduced as an option for the ISO JPEG XT reference implementation.[80]

There is persistent interest in encoding JPEG in unconventional ways that maximize image quality for a given file size. In 2014, Mozilla created **MozJPEG** from libjpeg-turbo, a slower but higher-quality encoder intended for web images.[81] In March 2017, Google released the open source project Guetzli, which trades off a much longer encoding time for smaller file size (similar to what Zopfli does for PNG and other lossless data formats).[82]

In April 2024, Google introduced **Jpegli**, a new JPEG coding library that offers enhanced capabilities and a 35% compression ratio improvement at high quality compression settings, while the coding speed is comparable with MozJPEG.[83]

Successors

The Joint Photographic Experts Group has developed several newer standards meant to complement or replace the functionality of the original JPEG format.

JPEG LS

Originating in 1993 and published as ISO-14495-1/ITU-T.87, JPEG LS offers a low-complexity lossless file format which was more efficient than JPEG's original lossless implementation. It also features a lossy mode close to lossless. Its functionality is largely limited to that, and largely shares the same limitations of the original JPEG in other aspects.

JPEG 2000

JPEG 2000 was published as ISO/IEC 15444 in December 2000. It is based on a discrete wavelet transform (DWT) and was designed to completely replace the original JPEG standard and exceed it in every way. It allows up to 38 bits per colour channel and 16384 channels, more than any other format, with a multitude of colour spaces, and thus high dynamic range (HDR). Furthermore, it supports alpha transparency coding, billions-by-billions pixel images, which is also more than any other format, and lossless compression. It has significantly improved lossy compression ratio with significantly less visible artefacts at strong compression levels.[84]

JPEG XT

JPEG XT (ISO/IEC 18477) was published in June 2015; it extends base JPEG format with support for higher integer bit depths (up to 16 bit), high dynamic range imaging and floating-point coding, lossless coding, and alpha channel coding. Extensions are backward compatible with the base JPEG/JFIF file format and 8-bit lossy compressed image. JPEG XT uses an extensible file format based on JFIF. Extension layers are used to modify the JPEG 8-bit base layer and restore the high-resolution image. Existing software is forward compatible and can read the JPEG XT binary stream, though it would only decode the base 8-bit layer.[85]

JPEG XL

JPEG XL (ISO/IEC 18181) was published in 2021–2022. It replaces the JPEG format with a new DCT-based royalty-free format and allows efficient transcoding as a storage option for traditional JPEG images.[86] The new format is designed to exceed the still image compression performance shown by HEIF HM, Daala and WebP. It supports billion-by-billion pixel images, up to 32-bit-per-component high dynamic range with the appropriate transfer functions (PQ and HLG), patch encoding of synthetic images such as bitmap fonts and gradients, animated images, alpha channel coding, and a choice of RGB/YCbCr/ICtCp color encoding.[87][88][89][90]

See also

- AVIF
- Better Portable Graphics, a format based on intra-frame encoding of the HEVC
- C-Cube, an early implementer of JPEG in chip form
- Comparison of graphics file formats
- Deblocking filter (video), the similar deblocking methods could be applied to JPEG
- Design rule for Camera File system (DCF)
- FELICS, a lossless image codec
- File extensions
- Graphics editing program
- High Efficiency Image File Format, image container format for HEVC and other image coding formats
- Lenna (test image), the traditional standard image used to test image processing algorithms
- Motion JPEG
- WebP

References

1. "T.81 – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES" (https://www.w3.org/Graphics/JPEG/itu-t81.pdf) (PDF). CCITT. September 1992. Archived (https://web.archive.org/web/20191230093239/http://www.w3.org/Graphics/JPEG/itu-t81.pdf) (PDF) from the original on 30 December 2019. Retrieved 12 July 2019.
2. "Definition of "JPEG" " (https://www.collinsdictionary.com/dictionary/english/jpeg). *Collins English Dictionary*. Archived (https://web.archive.org/web/20130921054412/http://www.collinsdictionary.com/dictionary/english/jpeg) from the original on 21 September 2013. Retrieved 23 May 2013.
3. "Overview of JPEG 1" (http://jpeg.org/jpeg). *Joint Photographic Experts Group*. Archived (https://web.archive.org/web/20250130141949/https://jpeg.org/jpeg/) from the original on 30 January 2025. Retrieved 5 February 2025.
4. Haines, Richard F.; Chuang, Sherry L. (1 July 1992). *The effects of video compression on acceptability of images for monitoring life sciences experiments* (https://ntrs.nasa.gov/search.jsp?R=19920024689) (Technical report). NASA. NASA-TP-3239, A-92040, NAS 1.60:3239. Retrieved 13 March 2016. "The JPEG still-image-compression levels, even with the large range of 5:1 to 120:1 in this study, yielded equally high levels of acceptability"

5. Hudson, Graham; Léger, Alain; Niss, Birger; Sebestyén, István; Vaaben, Jørgen (31 August 2018). "JPEG-1 standard 25 years: past, present, and future reasons for a success" (https://doi.org/10.1117%2F1.JEI.27.4.040901). *Journal of Electronic Imaging*. **27** (4): 1. doi:10.1117/1.JEI.27.4.040901 (https://doi.org/10.1117%2F1.JEI.27.4.040901). S2CID 52164892 (https://api.semanticscholar.org/CorpusID:52164892).
6. Svetlik, Joe (31 May 2018). "The JPEG Image Format Explained" (https://web.archive.org/web/20190805194553/https://home.bt.com/tech-gadgets/photography/what-is-a-jpeg-11364206889349). *BT.com*. BT Group. Archived from the original (https://home.bt.com/tech-gadgets/photography/what-is-a-jpeg-11364206889349) on 5 August 2019. Retrieved 5 August 2019.
7. Baraniuk, Chris (15 October 2015). "Copy Protections Could Come to JPEGs" (https://www.bbc.co.uk/news/technology-34538705). *BBC News*. BBC. Archived (https://web.archive.org/web/20191009193610/https://www.bbc.co.uk/news/technology-34538705) from the original on 9 October 2019. Retrieved 13 September 2019.
8. Trinkwalder, Andrea (7 October 2016). "JPEG: 25 Jahre und kein bisschen alt" (https://www.heise.de/newsticker/meldung/JPEG-25-Jahre-und-kein-bisschen-alt-3342519.html) [JPEG: 25 years (old) and not a bit old]. *de:Heise online* (in German). Archived (https://web.archive.org/web/20190905025359/https://www.heise.de/newsticker/meldung/JPEG-25-Jahre-und-kein-bisschen-alt-3342519.html) from the original on 5 September 2019. Retrieved 5 September 2019.
9. "T.81 – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES" (https://www.w3.org/Graphics/JPEG/itu-t81.pdf) (PDF). CCITT. September 1992. Retrieved 12 July 2019.
10. "JPEG: 25 Jahre und kein bisschen alt" (https://www.heise.de/newsticker/meldung/JPEG-25-Jahre-und-kein-bisschen-alt-3342519.html). *Heise online* (in German). October 2016. Retrieved 5 September 2019.
11. Caplan, Paul (24 September 2013). "What Is a JPEG? The Invisible Object You See Every Day" (https://www.theatlantic.com/technology/archive/2013/09/what-is-a-jpeg-the-invisible-object-you-see-every-day/279954/). *The Atlantic*. Archived (https://web.archive.org/web/20191009054159/https://www.theatlantic.com/technology/archive/2013/09/what-is-a-jpeg-the-invisible-object-you-see-every-day/279954/) from the original on 9 October 2019. Retrieved 13 September 2019.
12. "HTTP Archive – Interesting Stats" (http://httparchive.org/interesting.php#imageformats). *httparchive.org*. Retrieved 6 April 2016.
13. "MIME Type Detection in Internet Explorer" (https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms775147(v=vs.85)). Microsoft. 13 July 2016. Archived (https://web.archive.org/web/20221030181306/https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms775147(v=vs.85)) from the original on 30 October 2022. Retrieved 2 November 2022.
14. "JPEG File Interchange Format" (https://web.archive.org/web/20140903080533/http://www.jpeg.org/public/jfif.pdf) (PDF). 3 September 2014. Archived from the original on 3 September 2014. Retrieved 16 October 2017.
15. "Why JPEG 2000 Never Took Off" (https://blog.ansi.org/2018/07/why-jpeg-2000-never-used-standard-iso-iec/). *American National Standards Institute*. 10 July 2018. Archived (https://web.archive.org/web/20181216031019/https://blog.ansi.org/2018/07/why-jpeg-2000-never-used-standard-iso-iec/) from the original on 16 December 2018. Retrieved 13 September 2019.
16. Ahmed, Nasir (January 1991). "How I Came Up With the Discrete Cosine Transform" (https://www.scribd.com/doc/52879771/DCT-History-How-I-Came-Up-with-the-Discrete-Cosine-Transform). *Digital Signal Processing*. **1** (1): 4–5. doi:10.1016/1051-2004(91)90086-Z (https://doi.org/10.1016%2F1051-2004%2891%2990086-Z).

17. Ahmed, Nasir; Natarajan, T.; Rao, K. R. (January 1974), "Discrete Cosine Transform", *IEEE Transactions on Computers*, **C-23** (1): 90–93, doi:10.1109/T-C.1974.223784 (https://doi.org/10.1109%2FT-C.1974.223784), S2CID 149806273 (https://api.semanticscholar.org/CorpusID:149806273)
18. Lemos, Robert (23 July 2002). "Finding patent truth in JPEG claim" (https://www.cnet.com/news/finding-patent-truth-in-jpeg-claim/). *CNET*. Archived (https://web.archive.org/web/20190713165715/https://www.cnet.com/news/finding-patent-truth-in-jpeg-claim/) from the original on 13 July 2019. Retrieved 13 July 2019.
19. ISO/IEC JTC 1/SC 29 (7 May 2009). "ISO/IEC JTC 1/SC 29/WG 1 – Coding of Still Pictures (SC 29/WG 1 Structure)" (https://web.archive.org/web/20131231055215/http://kikaku.itscj.ipsj.or.jp/sc29/29w12901.htm). Archived from the original (http://kikaku.itscj.ipsj.or.jp/sc29/29w12901.htm) on 31 December 2013. Retrieved 11 November 2009.
20. ISO/IEC JTC 1/SC 29. "Programme of Work, (Allocated to SC 29/WG 1)" (https://web.archive.org/web/20131231012300/http://kikaku.itscj.ipsj.or.jp/sc29/29w42901.htm). Archived from the original (http://kikaku.itscj.ipsj.or.jp/sc29/29w42901.htm) on 31 December 2013. Retrieved 7 November 2009.
21. ISO. "JTC 1/SC 29 – Coding of audio, picture, multimedia and hypermedia information" (http://www.iso.org/iso/standards_development/technical_committees/list_of_iso_technical_committees/iso_technical_committee.htm?commid=45316). Archived (https://web.archive.org/web/20100703085542/http://www.iso.org/iso/standards_development/technical_committees/list_of_iso_technical_committees/iso_technical_committee.htm?commid=45316) from the original on 3 July 2010. Retrieved 11 November 2009.
22. JPEG. "Joint Photographic Experts Group, JPEG Homepage" (http://www.jpeg.org/jpeg/index.html). Archived (https://web.archive.org/web/20090927035043/http://www.jpeg.org/jpeg/index.html) from the original on 27 September 2009. Retrieved 8 November 2009.
23. "T.81: Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines" (http://www.itu.int/rec/T-REC-T.81). *Itu.int*. Archived (https://web.archive.org/web/20121106222712/http://www.itu.int/rec/T-REC-T.81) from the original on 6 November 2012. Retrieved 7 November 2009.
24. William B. Pennebaker; Joan L. Mitchell (1993). *JPEG still image data compression standard* (https://books.google.com/books?id=AepB_PZ_WMkC&dq=JPEG+%22did+not+specify+a+file+format%22&pg=PA291) (3rd ed.). Springer. p. 291. ISBN 978-0-442-01272-4.
25. ISO. "JTC 1/SC 29 – Coding of audio, picture, multimedia and hypermedia information" (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=45316). Archived (https://web.archive.org/web/20100703224947/http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=45316) from the original on 3 July 2010. Retrieved 7 November 2009.
26. "SPIFF, Still Picture Interchange File Format" (https://www.loc.gov/preservation/digital/formats/fdd/fdd000019.shtml). Library of Congress. 30 January 2012. Archived (https://web.archive.org/web/20180731093543/https://www.loc.gov/preservation/digital/formats/fdd/fdd000019.shtml) from the original on 31 July 2018. Retrieved 31 July 2018.
27. Louis Sharpe (24 April 2009). "JPEG XR enters FDIS status JPEG File Interchange Format (JFIF) to be standardized as JPEG Part 5" (https://web.archive.org/web/20091008041637/http://www.jpeg.org/newsrel25.html) (Press release). Archived from the original (http://www.jpeg.org/newsrel25.html) on 8 October 2009. Retrieved 9 November 2009.
28. "JPEG File Interchange Format (JFIF)" (http://www.ecma-international.org/publications/techreports/E-TR-098.htm). *ECMA TR/98 1st ed*. Ecma International. 2009. Archived (https://web.archive.org/web/20210114101520/http://www.ecma-international.org/publications/techreports/E-TR-098.htm) from the original on 14 January 2021. Retrieved 1 August 2011.
29. "Forgent's JPEG Patent" (https://pmt.sourceforge.io/SVG-patents/jpeg.html). *SourceForge*. 2002. Archived (https://web.archive.org/web/20190513043628/https://pmt.sourceforge.io/SVG-patents/jpeg.html) from the original on 13 May 2019. Retrieved 13 July 2019.

30. "Concerning recent patent claims" (https://web.archive.org/web/20070714232941/http://www.jpeg.org/newsrel1.html). *Jpeg.org*. 19 July 2002. Archived from the original (http://www.jpeg.org/newsrel1.html) on 14 July 2007. Retrieved 29 May 2011.
31. "JPEG and JPEG2000 – Between Patent Quarrel and Change of Technology" (https://web.archive.org/web/20040817154508/http://www.algovision-luratech.com/company/news/patentquarrel.jsp?OnlineShopId=164241031081525276). Archived from the original on 17 August 2004. Retrieved 16 April 2017.
32. Kawamoto, Dawn (22 April 2005). "Graphics patent suit fires back at Microsoft" (https://www.cnet.com/culture/graphics-patent-suit-fires-back-at-microsoft/). CNET News. Archived (https://web.archive.org/web/20230120140214/https://www.cnet.com/culture/graphics-patent-suit-fires-back-at-microsoft/) from the original on 20 January 2023. Retrieved 20 January 2023.
33. "Trademark Office Re-examines Forgent JPEG Patent" (http://arquivo.pt/wayback/20160515113624/http://www.publish.com/c/a/Graphics-Tools/Trademark-Office-Reexamines-Forgent-JPEG-Patent/). *Publish.com*. 3 February 2006. Archived from the original (http://www.publish.com/c/a/Graphics-Tools/Trademark-Office-Reexamines-Forgent-JPEG-Patent/) on 15 May 2016. Retrieved 28 January 2009.
34. "USPTO: Broadest Claims Forgent Asserts Against JPEG Standard Invalid" (http://www.groklaw.net/article.php?story=20060526105754880). *Groklaw.net*. 26 May 2006. Archived (https://web.archive.org/web/20190516201650/http://www.groklaw.net/article.php?story=20060526105754880) from the original on 16 May 2019. Retrieved 21 July 2007.
35. "Coding System for Reducing Redundancy" (https://web.archive.org/web/20110612114555/http://gauss.ffii.org/PatentView/EP266049). *Gauss.ffii.org*. Archived from the original (http://gauss.ffii.org/PatentView/EP266049) on 12 June 2011. Retrieved 29 May 2011.
36. "JPEG Patent Claim Surrendered" (http://www.pubpat.org/jpegsurrendered.htm). Public Patent Foundation. 2 November 2006. Archived (https://web.archive.org/web/20070102145054/http://www.pubpat.org/jpegsurrendered.htm) from the original on 2 January 2007. Retrieved 3 November 2006.
37. "Ex Parte Reexamination Certificate for U.S. Patent No. 5,253,341" (https://web.archive.org/web/20080602141045/http://www.uspto.gov/web/patents/patog/week30/OG/html/1320-4/US05253341-20070724.html). Archived from the original (http://www.uspto.gov/web/patents/patog/week30/OG/html/1320-4/US05253341-20070724.html) on 2 June 2008.
38. Workgroup. "Rozmanith: Using Software Patents to Silence Critics" (https://web.archive.org/web/20110716123228/http://eupat.ffii.org/pikta/xrani/rozmanith/index.en.html). *Eupat.ffii.org*. Archived from the original (http://eupat.ffii.org/pikta/xrani/rozmanith/index.en.html) on 16 July 2011. Retrieved 29 May 2011.
39. "A Bounty of $5,000 to Name Troll Tracker: Ray Niro Wants To Know Who Is saying All Those Nasty Things About Him" (http://www.law.com/jsp/article.jsp?id=1196762670106). *Law.com*. Archived (https://web.archive.org/web/20101121012911/http://www.law.com/jsp/article.jsp?id=1196762670106) from the original on 21 November 2010. Retrieved 29 May 2011.
40. Reimer, Jeremy (5 February 2008). "Hunting trolls: USPTO asked to reexamine broad image patent" (https://arstechnica.com/news.ars/post/20080205-hunting-trolls-uspto-asked-to-reexamine-broad-image-patent.html). *Arstechnica.com*. Archived (https://web.archive.org/web/20081208145656/http://arstechnica.com/news.ars/post/20080205-hunting-trolls-uspto-asked-to-reexamine-broad-image-patent.html) from the original on 8 December 2008. Retrieved 29 May 2011.
41. U.S. Patent Office – Granting Reexamination on 5,253,341 C1
42. "Judge Puts JPEG Patent On Ice" (https://www.techdirt.com/articles/20080427/143205960.shtml). *Techdirt.com*. 30 April 2008. Archived (https://web.archive.org/web/20111114020657/http://www.techdirt.com/articles/20080427/143205960.shtml) from the original on 14 November 2011. Retrieved 29 May 2011.

43. "JPEG Patent's Single Claim Rejected (And Smacked Down For Good Measure)" (https://techdirt.com/articles/20080731/0337491852.shtml). *Techdirt.com*. 1 August 2008. Archived (https://web.archive.org/web/20191128051716/https://www.techdirt.com/articles/20080731/0337491852.shtml) from the original on 28 November 2019. Retrieved 29 May 2011.
44. Workgroup. "Princeton Digital Image Corporation Home Page" (https://archive.today/20130411084058/http://www.princetondigitalimage.com/). Archived from the original (http://www.princetondigitalimage.com/) on 11 April 2013. Retrieved 1 May 2013.
45. Workgroup (3 April 2013). "Article on Princeton Court Ruling Regarding GE License Agreement" (https://web.archive.org/web/20160309142416/http://patentlaw.jmbm.com/2013/04/hps-motion-to-dismiss-for-lack.html). Archived from the original (http://patentlaw.jmbm.com/2013/04/hps-motion-to-dismiss-for-lack.html/) on 9 March 2016. Retrieved 1 May 2013.
46. Kaur, Rajandeep (May 2016). "A Review of Image Compression Techniques" (https://www.researchgate.net/publication/303325238). *International Journal of Computer Applications*. **142** (1): 8–11. doi:10.5120/ijca2016909658 (https://doi.org/10.5120%2Fijca2016909658).
47. "Progressive Decoding Overview" (http://msdn.microsoft.com/en-us/library/ee720036(v=vs.85).aspx). *Microsoft Developer Network*. Microsoft. Archived (https://web.archive.org/web/20121119054723/http://msdn.microsoft.com/en-us/library/ee720036(v=vs.85).aspx) from the original on 19 November 2012. Retrieved 23 March 2012.
48. Fastvideo (May 2019). "12-bit JPEG encoder on GPU" (https://www.fastcompression.com/products/12-bit-jpeg-codec.htm). Archived (https://web.archive.org/web/20190506142226/https://www.fastcompression.com/products/12-bit-jpeg-codec.htm) from the original on 6 May 2019. Retrieved 6 May 2019.
49. "Why You Should Always Rotate Original JPEG Photos Losslessly" (http://petapixel.com/2012/08/14/why-you-should-always-rotate-original-jpeg-photos-losslessly/). *Petapixel.com*. 14 August 2012. Archived (https://web.archive.org/web/20171017093502/https://petapixel.com/2012/08/14/why-you-should-always-rotate-original-jpeg-photos-losslessly/) from the original on 17 October 2017. Retrieved 16 October 2017.
50. "JFIF File Format as PDF" (http://www.w3.org/Graphics/JPEG/jfif3.pdf) (PDF). Archived (https://web.archive.org/web/20210113062300/http://www.w3.org/Graphics/JPEG/jfif3.pdf) (PDF) from the original on 13 January 2021. Retrieved 19 June 2006.
51. Tom Lane (29 March 1999). "JPEG image compression FAQ" (http://www.faqs.org/faqs/jpeg-faq/part1/). Archived (https://web.archive.org/web/20101110054244/http://www.faqs.org/faqs/jpeg-faq/part1/) from the original on 10 November 2010. Retrieved 11 September 2007. (q. 14: "Why all the argument about file formats?")
52. "Everything you need to know about JPEG files | Adobe" (https://www.adobe.com/creativecloud/file-types/image/raster/jpeg-file.html). *www.adobe.com*. Retrieved 18 August 2023.
53. "A Standard Default Color Space for the Internet - sRGB" (https://www.w3.org/Graphics/Color/sRGB.html). *www.w3.org*. Archived (https://web.archive.org/web/20220218083217/https://www.w3.org/Graphics/Color/sRGB.html) from the original on 18 February 2022. Retrieved 18 February 2022.
54. "IEC 61966-2-1:1999/AMD1:2003 | IEC Webstore" (https://webstore.iec.ch/publication/6168). *webstore.iec.ch*. Archived (https://web.archive.org/web/20220218083451/https://webstore.iec.ch/publication/6168) from the original on 18 February 2022. Retrieved 18 February 2022.
55. "ISO/IEC 10918-1: 1993(E) p.36" (http://www.digicamsoft.com/itu/itu-t81-36.html). Archived (https://web.archive.org/web/20110801031604/http://www.digicamsoft.com/itu/itu-t81-36.html) from the original on 1 August 2011. Retrieved 30 November 2007.
56. Thomas G. Lane. "Advanced Features: Compression parameter selection" (http://apodeline.free.fr/DOC/libjpeg/libjpeg-3.html). *Using the IJG JPEG Library*. Archived (https://web.archive.org/web/20011126012948/http://apodeline.free.fr/DOC/libjpeg/libjpeg-3.html) from the original on 26 November 2001. Retrieved 8 October 2008.

57. Ryan, Dan (20 June 2012). *E - Learning Modules: Dlr Associates Series* (https://books.google.com/books?id=bYxMVVzdV80C&dq=A+particular+conversion+to+Y%E2%80%B2CBCR+is+specified+in+the+JFIF+standard%2C+and+should+be+performed+for+the+resulting+JPEG+file+to+have+maximum+compatibility.+A+particular+conversion+to+Y%E2%80%B2CBCR+is+specified+in+the+JFIF+standard%2C+and+should+be+performed+for+the+resulting+JPEG+file+to+have+maximum+compatibility.++A+particular+conversion+to+Y%E2%80%B2CBCR+is+specified+in+the+JFIF+standard%2C+and+should+be+performed+for+the+resulting+JPEG+file+to+have+maximum+compatibility.+&pg=PA53). AuthorHouse. ISBN 978-1-4685-7520-0.
58. "DC / AC Frequency Questions - Doom9's Forum" (http://forum.doom9.org/showthread.php?p=184647#post184647). *forum.doom9.org*. Archived (https://web.archive.org/web/20171017042422/http://forum.doom9.org/showthread.php?p=184647#post184647) from the original on 17 October 2017. Retrieved 16 October 2017.
59. Maini, Raman; Mehra, Suruchi (December 2010). "A Review on JPEG2000 Image Compression" (https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=bcdc5083bcf5c3120e7450273bd380f61071afcb). *International Journal of Computer Applications*. **11** (9): 43–47. doi:10.5120/1607-2159 (https://doi.org/10.5120%2F1607-2159) – via CiteSeerX.
60. Phuc-Tue Le Dinh and Jacques Patry. Video compression artifacts and MPEG noise reduction (http://www.videsignline.com/howto/180207350) Archived (https://web.archive.org/web/20060314201405/http://www.videsignline.com/howto/180207350) 2006-03-14 at the Wayback Machine. Video Imaging DesignLine. February 24, 2006. Retrieved May 28, 2009.
61. "**3.9 mosquito noise:** Form of edge busyness distortion sometimes associated with movement, characterized by moving artifacts and/or blotchy noise patterns superimposed over the objects (resembling a mosquito flying around a person's head and shoulders)." ITU-T Rec. P.930 (08/96) Principles of a reference impairment system for video (http://eu.sabotage.org/www/ITU/P/P0930e.pdf) Archived (https://web.archive.org/web/20100216033245/http://eu.sabotage.org/www/ITU/P/P0930e.pdf) 2010-02-16 at the Wayback Machine
62. Julià Minguillón, Jaume Pujol (April 2001). "JPEG standard uniform quantization error modeling with applications to sequential and progressive operation modes" (http://openaccess.uoc.edu/webapps/o2/bitstream/10609/6263/6/jei-jpeg.pdf) (PDF). *Electronic Imaging*. **10** (2): 475–485. Bibcode:2001JEI....10..475M (https://ui.adsabs.harvard.edu/abs/2001JEI....10..475M). doi:10.1117/1.1344592 (https://doi.org/10.1117%2F1.1344592). hdl:10609/6263 (https://hdl.handle.net/10609%2F6263). S2CID 16629522 (https://api.semanticscholar.org/CorpusID:16629522). Archived (https://web.archive.org/web/20200803042010/http://openaccess.uoc.edu/webapps/o2/bitstream/10609/6263/6/jei-jpeg.pdf) (PDF) from the original on 3 August 2020. Retrieved 23 September 2019.
63. I. Bauermann and E. Steinbacj. Further Lossless Compression of JPEG Images. Proc. of Picture Coding Symposium (PCS 2004), San Francisco, US, December 15–17, 2004.
64. N. Ponomarenko, K. Egiazarian, V. Lukin and J. Astola. Additional Lossless Compression of JPEG Images, Proc. of the 4th Intl. Symposium on Image and Signal Processing and Analysis (ISPA 2005), Zagreb, Croatia, pp. 117–120, September 15–17, 2005.
65. M. Stirner and G. Seelmann. Improved Redundancy Reduction for JPEG Files. Proc. of Picture Coding Symposium (PCS 2007), Lisbon, Portugal, November 7–9, 2007
66. Ichiro Matsuda, Yukio Nomoto, Kei Wakabayashi and Susumu Itoh. Lossless Re-encoding of JPEG images using block-adaptive intra prediction. Proceedings of the 16th European Signal Processing Conference (EUSIPCO 2008).
67. Stirner, Matthias (19 February 2023). "packjpg/packJPG" (https://github.com/packjpg/packJPG). *GitHub*. Archived (https://web.archive.org/web/20230302013300/https://github.com/packjpg/packJPG) from the original on 2 March 2023. Retrieved 2 March 2023.
68. J. Siragusa; D. C. Swift (1997). "General Purpose Stereoscopic Data Descriptor" (https://web.archive.org/web/20111030182549/http://vrex.com/developer/sterdesc.pdf) (PDF). *VRex, Inc., Elmsford, New York, US*. Archived from the original (http://vrex.com/developer/sterdesc.pdf) (PDF) on 30 October 2011.

69. Tim Kemp, JPS files (http://ephehm.com/jps/) Archived (https://web.archive.org/web/20090118200346/http://ephehm.com/jps/) 2009-01-18 at the Wayback Machine
70. "CGImageSource.SupportedTypes" (https://www.mbsplugins.eu/CGImageSourceSupportedTypes.shtml). *Claris FileMaker MBS Plug-in*. MonkeyBread Software. Archived (https://web.archive.org/web/20201230132850/https://mbsplugins.eu/CGImageSourceSupportedTypes.shtml) from the original on 30 December 2020. Retrieved 21 May 2023.
71. "Multi-Picture Format" (https://web.archive.org/web/20160405200235/http://cipa.jp/std/documents/e/DC-007_E.pdf) (PDF). 2009. Archived from the original (http://www.cipa.jp/std/documents/e/DC-007_E.pdf) (PDF) on 5 April 2016. Retrieved 30 December 2015.
72. "MPO2Stereo: Convert Fujifilm MPO files to JPEG stereo pairs" (http://www.mtbs3d.com/phpbb/viewtopic.php?f=3&t=4124&start=15), *Mtbs3d.com*, archived (https://web.archive.org/web/20100531020644/http://www.mtbs3d.com/phpbb/viewtopic.php?f=3&t=4124&start=15) from the original on 31 May 2010, retrieved 12 January 2010
73. Alessandro Ortis; Sebastiano Battiato (2015), Sitnik, Robert; Puech, William (eds.), "A new fast matching method for adaptive compression of stereoscopic images" (http://iplab.dmi.unict.it/publication/503), *Three-Dimensional Image Processing*, Three-Dimensional Image Processing, Measurement (3DIPM), and Applications 2015, **9393**, SPIE - Three-Dimensional Image Processing, Measurement (3DIPM), and Applications 2015: 93930K, Bibcode:2015SPIE.9393E..0KO (https://ui.adsabs.harvard.edu/abs/2015SPIE.9393E..0KO), doi:10.1117/12.2086372 (https://doi.org/10.1117%2F12.2086372), S2CID 18879942 (https://api.semanticscholar.org/CorpusID:18879942), archived (https://web.archive.org/web/20160303232457/http://iplab.dmi.unict.it/publication/503) from the original on 3 March 2016, retrieved 30 April 2015
74. Alessandro Ortis; Francesco Rundo; Giuseppe Di Giore; Sebastiano Battiato, *Adaptive Compression of Stereoscopic Images* (http://iplab.dmi.unict.it/publication/49), International Conference on Image Analysis and Processing (ICIAP) 2013, archived (https://web.archive.org/web/20160303205755/http://iplab.dmi.unict.it/publication/49) from the original on 3 March 2016, retrieved 30 April 2015
75. Tom Lane, January 16, 2013: jpeg-9, API/ABI compatibility, and the future role of this project (https://lists.fedoraproject.org/pipermail/devel/2013-January/176400.html) Archived (https://web.archive.org/web/20181204231539/https://lists.fedoraproject.org/pipermail/devel/2013-January/176400.html) 2018-12-04 at the Wayback Machine
76. Software That Uses or Provides libjpeg-turbo (http://libjpeg-turbo.virtualgl.org/About/Software) Archived (https://web.archive.org/web/20170318000736/http://libjpeg-turbo.virtualgl.org/About/Software) 2017-03-18 at the Wayback Machine. February 9, 2012.
77. Issue 48789 – chromium – Use libjpeg-turbo instead of libjpeg (https://code.google.com/p/chromium/issues/detail?id=48789#c19) Archived (https://web.archive.org/web/20150801035040/http://code.google.com/p/chromium/issues/detail?id=48789#c19) 2015-08-01 at the Wayback Machine. April 14, 2011.
78. "ISO/IEC 10918-7: 2019 Information technology — Digital compression and coding of continuous-tone still images — Part 7: Reference software" (https://www.iso.org/standard/75845.html). *ISO*. Archived (https://web.archive.org/web/20220507152432/https://www.iso.org/standard/75845.html) from the original on 7 May 2022. Retrieved 7 May 2022."T.873 (05/19): Information technology - Digital compression and coding of continuous-tone still images: Reference software" (https://www.itu.int/rec/T-REC-T.873-201905-S/en). *www.itu.int*. Archived (https://web.archive.org/web/20220602173200/https://www.itu.int/rec/T-REC-T.873-201905-S/en) from the original on 2 June 2022. Retrieved 1 March 2023.
79. "JPEG - JPEG XT" (https://jpeg.org/jpegxt/software.html). *jpeg.org*. Archived (https://web.archive.org/web/20180304055159/https://jpeg.org/jpegxt/software.html) from the original on 4 March 2018. Retrieved 3 March 2018.

80. Richter, Thomas (September 2016). "JPEG on STEROIDS: Common optimization techniques for JPEG image compression". *2016 IEEE International Conference on Image Processing (ICIP)*. pp. 61–65. doi:10.1109/ICIP.2016.7532319 (https://doi.org/10.1109%2FICIP.2016.7532319). ISBN 978-1-4673-9961-6. S2CID 14922251 (https://api.semanticscholar.org/CorpusID:14922251).
81. "Introducing the 'mozjpeg' Project" (https://research.mozilla.org/2014/03/05/introducing-the-mozjpeg-project/). *Mozilla Research*. 5 March 2014. Archived (https://web.archive.org/web/20230301034809/https://research.mozilla.org/2014/03/05/introducing-the-mozjpeg-project/) from the original on 1 March 2023. Retrieved 1 March 2023.
82. "Announcing Guetzli: A New Open Source JPEG Encoder" (https://research.googleblog.com/2017/03/announcing-guetzli-new-open-source-jpeg.html). *Research.googleblog.com*. 16 March 2017. Archived (https://web.archive.org/web/20171006012117/https://research.googleblog.com/2017/03/announcing-guetzli-new-open-source-jpeg.html) from the original on 6 October 2017. Retrieved 16 October 2017.
83. "Introducing Jpegli: A New JPEG Coding Library" (https://web.archive.org/web/20240403181607/https://opensource.googleblog.com/2024/04/introducing-jpegli-new-jpeg-coding-library.html). Google Open Source Blog. 3 April 2024. Archived from the original (https://opensource.googleblog.com/2024/04/introducing-jpegli-new-jpeg-coding-library.html) on 3 April 2024. Retrieved 4 April 2024.
84. Sneyers, Jon (22 February 2021). "It's High Time to Replace JPEG With a Next-Generation Image Codec" (https://cloudinary.com/blog/time_for_next_gen_codecs_to_dethrone_jpeg). *Cloudinary*. Retrieved 14 November 2023.
85. "JPEG - JPEG XT" (https://jpeg.org/jpegxt/). *jpeg.org*. Archived (https://web.archive.org/web/20180304055205/https://jpeg.org/jpegxt/) from the original on 4 March 2018. Retrieved 3 March 2018.
86. Alakuijala, Jyrki; van Asseldonk, Ruud; Boukortt, Sami; Bruse, Martin; Comșa, Iulia-Maria; Firsching, Moritz; Fischbacher, Thomas; Kliuchnikov, Evgenii; Gomez, Sebastian; Obryk, Robert; Potempa, Krzysztof; Rhatushnyak, Alexander; Sneyers, Jon; Szabadka, Zoltan; Vandervenne, Lode; Versari, Luca; Wassenberg, Jan (6 September 2019). "JPEG XL next-generation image compression architecture and coding tools" (https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11137/111370K/JPEG-XL-next-generation-image-compression-architecture-and-coding-tools/10.1117/12.2529237.full?SSO=1). In Tescher, Andrew G; Ebrahimi, Touradj (eds.). *Applications of Digital Image Processing XLII*. Vol. 11137. p. 20. Bibcode:2019SPIE11137E..0KA (https://ui.adsabs.harvard.edu/abs/2019SPIE11137E..0KA). doi:10.1117/12.2529237 (https://doi.org/10.1117%2F12.2529237). ISBN 978-1-5106-2967-7. S2CID 202785129 (https://api.semanticscholar.org/CorpusID:202785129). Archived (https://web.archive.org/web/20211226010413/https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11137/111370K/JPEG-XL-next-generation-image-compression-architecture-and-coding-tools/10.1117/12.2529237.full?SSO=1) from the original on 26 December 2021. Retrieved 26 December 2021.
87. Rhatushnyak, Alexander; Wassenberg, Jan; Sneyers, Jon; Alakuijala, Jyrki; Vandevenne, Lode; Versari, Luca; Obryk, Robert; Szabadka, Zoltan; Kliuchnikov, Evgenii; Comsa, Iulia-Maria; Potempa, Krzysztof; Bruse, Martin; Firsching, Moritz; Khasanova, Renata; Ruud van Asseldonk; Boukortt, Sami; Gomez, Sebastian; Fischbacher, Thomas (2019). "Committee Draft of JPEG XL Image Coding System". arXiv:1908.03565 (https://arxiv.org/abs/1908.03565) [eess.IV (https://arxiv.org/archive/eess.IV)].
88. "N79010 Final Call for Proposals for a Next-Generation Image Coding Standard (JPEG XL)" (https://jpeg.org/downloads/jpegxl/jpegxl-cfp.pdf) (PDF). *ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG16)*. Archived (https://web.archive.org/web/20221031184400/https://jpeg.org/downloads/jpegxl/jpegxl-cfp.pdf) (PDF) from the original on 31 October 2022. Retrieved 29 May 2018.
89. *ISO/IEC 18181-1:2022 Information technology — JPEG XL image coding system — Part 1: Core coding system* (https://www.iso.org/standard/77977.html).

90. *ISO/IEC 18181-2:2021 Information technology — JPEG XL image coding system — Part 2: File format* (https://www.iso.org/standard/80617.html).

External links

- Official website (https://jpeg.org/jpeg/)
- JPEG Standard (JPEG ISO/IEC 10918-1 ITU-T Recommendation T.81) (https://www.w3.org/Graphics/JPEG/itu-t81.pdf) at W3.org
- JPEG File Interchange Format (JFIF), Version 1.02, Sept. 1992 (https://www.w3.org/Graphics/JPEG/jfif3.pdf) at W3.org
- Format description of JPEG Image Coding Family (https://www.loc.gov/preservation/digital/formats/fdd/fdd000017.shtml) from Library of Congress
- Example images over the full range of quantization levels from 1 to 100 (https://www.visengi.com/products/jpeg_hardware_encoder) at visengi.com

Retrieved from "https://en.wikipedia.org/w/index.php?title=JPEG&oldid=1289330902"